VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Joonas Höykinpuro

IO-LISTA-KOODIGENERAATTORI

Tekniikka
2020

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Joonas Höykinpuro
Opinnäytetyön nimi	IO-lista-koodigeneraattori
Vuosi	2020
Kieli	suomi
Sivumäärä	21
Ohjaaja	Timo Kankaanpää

Tämän opinnäytetyön tavoitteena oli luoda toimeksiantajayritys VEO Oy:lle ohjelma, jonka avulla voidaan tehokkaasti generoida logiikan toimilohkoja kuvaavat tiedostot Microsoft Excel -pohjaisesta IO-listasta. Sovelluksen generoimat tiedostot voidaan helposti tuoda logiikkaohjelmointiin tarkoitettuun ohjelmistoon import-toiminnolla, mikä helpottaa ja nopeuttaa PLC-koodin tekemistä huomattavasti.

Työn toteutuksessa käytettiin Visual Basic .NET-ohjelmointikieltä sekä Microsoft Visual Studio 2017-ohjelmointiympäristöä. Ohjelman generoima tiedosto on XML-muotoinen tiedosto. PLC-ohjelmointiohjelma, johon generoitu tiedosto voidaan tuoda, on EcoStruxure Control Expert.

Työn tuloksena syntynyt ohjelma täytti kaikki sille asetetut vaatimukset ja kehitetyn ohjelman avulla voidaan generoida yhteensä 34 eri toimilohkokaaviota kuvaavaa XML-tiedostoa. Ohjelmalla generoitujen tiedostojen yksilöllinen sisältö määräytyy generoinnissa käytettävän IO-listan sisällön perusteella.

Avainsanat automaatio, PLC, ohjelmointi, VB.NET

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Joonas Höykinpuro
Title	IO List Code Generator
Year	2020
Language	Finnish
Pages	21
Name of Supervisor	Timo Kankaanpää

The purpose of this thesis was to implement a software for the client company VEO Oy. The goal of the implemented software was to efficiently generate files describing PLC function blocks from a Microsoft Excel based IO list. The files that the software generates can be effortlessly imported to the PLC programming software, which makes the PLC programming process significantly easier and faster.

Tools and techniques used for the development of the software were Visual Basic .NET programming language and Microsoft Visual Studio 2017 integrated development environment. The file type of the generated file that the software generates is XML file. PLC programming software to which the generated file can be imported into is EcoStruxure Control Expert.

The software that was developed during this project met all the requirements which were set for it. Developed software can be used to generate total of 34 different files which describe function block diagrams in XML format. The unique content of generated files is determined by the content of the IO-list used for the generation process.

Keywords automation, PLC, software development, VB.NET

SISÄLLYS

LYHENNELUETTELO

FBD	Function Block Diagram, ohjelmoitavan logiikan ohjelmointikieli
IL	Instruction List, ohjelmoitavan logiikan ohjelmointikieli
IO	Input/Output, PLC-järjestelmän sisään- ja ulostulot
LD	Ladder Diagram, ohjelmoitavan logiikan ohjelmointikieli
PLC	Programmable Logic Controller, ohjelmoitava logiikka
SFC	Sequential Function Chart, ohjelmoitavan logiikan ohjelmointikieli
ST	Structured Text, ohjelmoitavan logiikan ohjelmointikieli
XML	Extensible Markup Language, merkintäkieli
VB.NET	Visual Basic .NET, ohjelmointikieli
XAML	Extensible Application Markup Language, graafisen käyttöliittymän kuvaava merkintäkieli
WPF	Windows Presentation Foundation, graafisen käyttöliittymän sovelluskehys

KUVIO- JA TAULUKKOLUETTELO

1 JOHDANTO

1.1 VEO Oy

Tämän opinnäytetyön toimeksiantajana oli vaasalainen yritys VEO Oy. VEO Oy on perustettu Vaasassa nimellä Vaasa Engineering Oy vuonna 1989. VEO Oy on energiateollisuuden yritys ja yrityksen pääkonttori sijaitsee Vaasan Runsorissa. Se oli ensimmäinen Runsorin alueelle toimintansa sijoittanut energiateollisuuden yritys /1/.

Yrityksen tuotevalikoimaan kuuluu automaatio- ja sähkönjakeluratkaisujen tarjoaminen energia- ja prosessiteollisuuden yrityksille. Yrityksen tarjoamien järjestelmien käyttökohteita voivat olla esimerkiksi voimalaitokset, prosessiteollisuuden yritykset, laivat sekä nosturit /2/.

1.2 Työn tarve

Toimeksiantajayritys VEO Oy:llä oli tarve ohjelmalle, jonka avulla automaatioon liittyvää logiikkaohjelmointiprosessia voidaan nopeuttaa ja helpottaa automatisoimalla FBD-pohjaisten logiikkaohjelmien toimilohkojen luominen sekä toimilohkojen sisään- ja ulostulojen automaattinen kytkeminen. Toimilohkojen luonti tapahtuu projektikohtaisen IO-listan perusteella, jossa määritellään projektissa tarvittavat sisään- ja ulostulot.

PLC-ohjelman tekemisen toistuviin työvaiheisiin, kuten itse toimilohkon luomiseen ja sen nastojen kytkemiseen kuluu helposti paljon aikaa. Työssä tehtävän ohjelman avulla nämä työvaiheet voidaan hoitaa ohjelmallisesti sen sijaan, että PLC-ohjelmaa tekevän työntekijän täytyisi tehdä ne jokaisessa projektissa erikseen.

1.3 PLC lyhyesti

PLC eli ohjelmoitava logiikka on automaatiossa käytettävä pieni tietokone, jonka avulla voidaan automatisoida erilaisten koneiden ohjaus. PLC-järjestelmien käyttökohteita voivat olla esimerkiksi teollisuudessa käytettävät monimutkaiset koneet tai jopa yksinkertaiset laitteet kuten autotallin oven avaajat /3/.

PLC koostuu sisään- ja ulostuloista sekä prosessorista. Ohjelmoitavan logiikan toiminta perustuu siihen kytkettyjen sisääntulojen tilan tarkasteluun. Sisääntulojen tilan perusteella PLC:n prosessorin muistiin tallennettu ohjelma määrittää ulostuloihin kytkettyjen laitteiden tilan (Kuvio 1). Ohjelmoitavassa logiikassa yleisimmin käytetyt ohjelmointikielet ovat LD, FBD, ST, IL sekä SFC /3/.

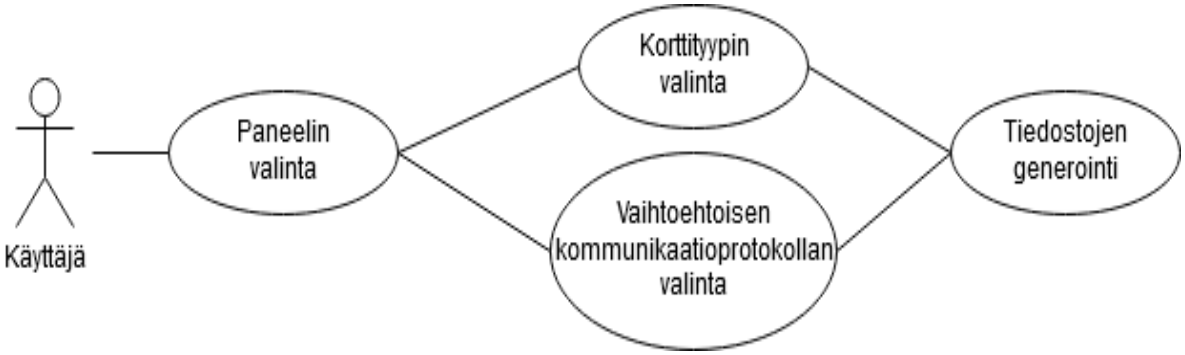
Kuvio 1. Esimerkkejä ohjelmoitavan logiikan sisään- ja ulostuloihin kytkettävistä laitteista /3/.

2 MÄÄRITTELY

2.1 Yleiskuvaus

IO-lista-koodigeneraattori on ohjelma, joka voidaan lisätä Microsoft Exceliin kolmannen osapuolen Add-in -apuohjelmana. Kerran lisätty apuohjelma on käytettävissä Excelissä, kunnes se poistetaan manuaalisesti. IO-lista-koodigeneraattorin käynnistysnappi lisätään mukautettuun välilehteen, jossa on valmiiksi jo muita VEO Oy:n käyttämiä IO-listan käsittelyyn kehitettyjä makroja sekä sovelluksia. Käynnistyessään ohjelma käyttää avoinna olevaa Excel-tiedostoa automaattisesti lähdetiedostona, joten erillistä tiedoston valintaominaisuutta ei tarvita, kun ohjelma käynnistetään avoinna olevasta IO-listasta.

Ohjelman käynnistäminen avaa graafisen käyttöliittymän, jossa käyttäjä voi valita haluamansa paneelit sekä tehdä muut valinnat. Kun halutut valinnat on tehty, voidaan nappia painamalla generoida IO-listan perusteella XML-tiedosto, jossa kuvataan toimilohkokaavion rakenne, toimilohkot sekä toimilohkoihin kytketyt sisään- ja ulostulot. Tämän jälkeen generoitu tiedosto voidaan tuoda EcoStruxure Control Expert PLC-ohjelmointiohjelmaan sen import-ominaisuudella. Toimijan mahdolliset toiminnot kuvattu alla olevassa ohjelman käyttötapauskaaviossa (Kuvio 2).



Kuvio 2. Ohjelman käyttötapauskaavio.

2.2 Toiminnot

Alla olevassa taulukossa kuvattu ohjelman toiminnot sekä niiden tärkeysasteet. Tärkeysasteita kuvataan numeroilla 1–3. Numero 1 tarkoittaa toimintoa, joka täytyy toteutua, numero 3 toimintoa, joka olisi kiva olla.

Taulukko 1. Sovelluksen vaatimukset ja niiden tärkeysasteet.

Viite	Vaatimuksen kuvaus	Tärkeys
T1	Logiikan toimilohkojen generointi	1
T2	Logiikan toimilohkojen IO-nastojen kytkeminen	1
T3	Graafinen käyttöliittymä	1
T4	Sovelluksen suorituksenaikaisten virheiden kirjoitus tiedostoon	2
T5	Haluttujen paneelien valinta	1
T6	Toimilohkojen sijainnin dynaaminen laskeminen	2
T7	Toimilohkojen selityksen generointi	3
T8	Haluttujen korttityyppien valinta	1
T9	Vaihtoehtoisen kommunikaatioprotokollan valinta	1
T10	Paneelikohtaisten valintaruutujen valitsemisen salliminen tai estäminen riippuen valitusta paneelista	3

3 SUUNNITTELU JA TOTEUTUS

3.1 Käytetyt tekniikat ja kehitystyökalut

Ohjelmointikieleksi IO-lista-koodigeneraattorin toteutukseen valikoitui Visual Basic .NET -ohjelmointikieli, sillä VB.NET tarjoaa luokkakirjaston, joka sisältää valmiita keinoja Microsoft Excel -tiedostojen käsittelyyn. Tämän lisäksi VB.NET-ohjelmointikielellä on toteutettu muitakin VEO Oy:n IO-listan käsittelyyn tarkoitettuja ohjelmia, joten senkin puolesta VB.NET-ohjelmointikielen valinta oli selvä. Graafinen käyttöliittymä toteutettiin käyttäen XAML-pohjaista WPF-sovelluskehystä.

Kehitystyökaluna ohjelmoinnissa käytettiin Microsoft Visual Studio 2017 -ohjelmointiympäristöä. Kehityksessä apuna käytetyt kirjastot sekä niiden käyttötarkoitus kehitetyssä ohjelmassa lueteltuna alla:

- System.Configuration
  - Sovellusasetuksien lukeminen konfigurointitiedostosta
- System.IO
  - Hakemisto- ja tiedostonkäsittely, datavirran kirjoitus tiedostoon
- System.Xml
  - XML-tiedostojen käsittely
- Microsoft.Office.Interop.Excel
  - Microsoft Excel -tiedostojen käsittely
- System.Windows
  - Käyttöliittymän tapahtumien käsittely
- System.Windows.Input
  - Käyttöliittymän kursorin ulkoasun ohjaus

3.1.1 Visual Basic .NET

VB.NET eli Visual Basic .NET on Microsoftin vuonna 2002 julkaisema ohjelmointikieli, jolla korvattiin Visual Basic 6 -ohjelmointikieli. VB.NET on suunni-

teltu helposti ymmärrettäväksi ohjelmointikieleksi sekä aloitteleville, että jo kokeneemmille ohjelmoijille. VB.NET on toteutettu .NET-sovelluskehyksen päälle, joten sillä pystyy käyttämään kaikkia .NET-sovelluskehyksen sisältämiä kirjastoja. .NET-sovelluskehyksen ansiosta VB.NET-ohjelmointikielellä kehitettyjen ohjelmien suoritukset ovat luotettavia sekä ohjelmat hyvin skaalautuvia. VB.NET on moniparadigmainen ohjelmointikieli ja se tukee myös olio-ohjelmointia /4/.

3.1.2 Windows Presentation Foundation

WPF eli Windows Presentation Foundation on Microsoftin kehittämä .NET-sovelluskehyksen kanssa käytettävä graafisen käyttöliittymän kehittämiseen tarkoitettu sovelluskehys. WPF-sovelluskehyksen avulla kehitetyt graafiset käyttöliittymät perustuvat XAML-merkintäkieleen /5/.

WPF-sovelluskehyksen avulla pystytään luoda sovellukseen helposti elementit, joista graafinen käyttöliittymä koostuu. Esimerkkejä näistä elementeistä on muun muassa nimikentät, alasvetovalikot, valintaruudut sekä tekstikentät /6/.

3.2 Vaatimuksien siirtäminen ohjelmistoympäristöön

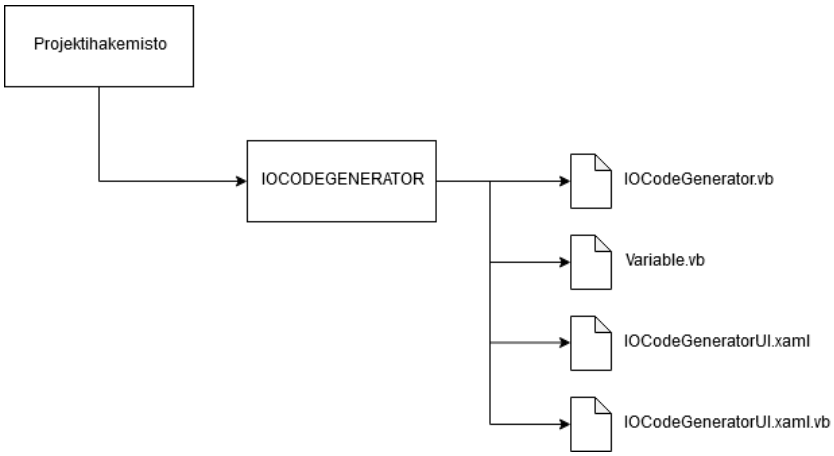
Ennen varsinaisen ohjelman kehitysprosessin alkamista aiheesta pidettiin aloituspalaveri, jossa käytiin läpi, miten kehitettävän ohjelman tulee toimia sekä minkälaisia toimintoja siltä vaaditaan. Alkupalaverissa päätettiin myös ohjelman kehityksessä käytettävät tekniikat sekä kehitystyökalut. Palaverit jatkuivat säännöllisesti koko ohjelman kehitysprosessin ajan. Palavereissa käytiin läpi muun muassa, mitä uutta ohjelmaan on tullut viime palaverin jälkeen sekä keskusteltiin mahdollisista toimintojen lisäyksistä, muutoksista tai hylkäämisistä.

Itse ohjelmointiprosessin kulkua ei toimeksiantajayrityksen puolesta määrätty, vaan minulla oli vapaus toteuttaa ohjelma parhaaksi näkemälläni tavalla käytettävillä tekniikoilla ja työkaluilla siten, että ohjelmalta vaaditut toiminnot ja ominaisuudet saavutetaan. Ohjelman kehityksessä lähdettiin liikkeelle siten, että ohjelma jaettiin osiin vaadittujen toimintojen perusteella ja ohjelmakoodi kirjoitettiin aina yksi osa kerrallaan. Ohjelma pyrittiin toteuttamaan siten, että ohjelman eri osat ovat erillään muista osista omissa aliohjelmissaan.

Ohjelmointiprosessi aloitettiin siten, että aluksi tehtiin moduuli, joka luo XML-tiedostoon toimilohkokaavion pohjan, joka on aina lähes samanlainen IO-listan tiedoista riippumatta, vain tiedostonimi muuttuu. Seuraavaksi tehtiin graafisen käyttöliittymän karkea hahmotelma, jossa pystyi tekemään haluttuja valintoja sekä osat, joiden avulla luotuun toimilohkokaavion pohjaan lisätään IO-listan sisällön perusteella käyttöliittymässä tehtyjen valintojen mukaiset yksilökohtaiset toimilohkot kokonaisuudessaan. Ohjelman toiminnallisten osion valmistuttua ryhdyttiin viimeistelemään graafista käyttöliittymää, sillä ajatuksella, että se pysyisi yksinkertaisena ja helppokäyttöisenä, eikä sen käyttöön tarvitsisi erillisiä ohjeita tai käyttökoulutusta.

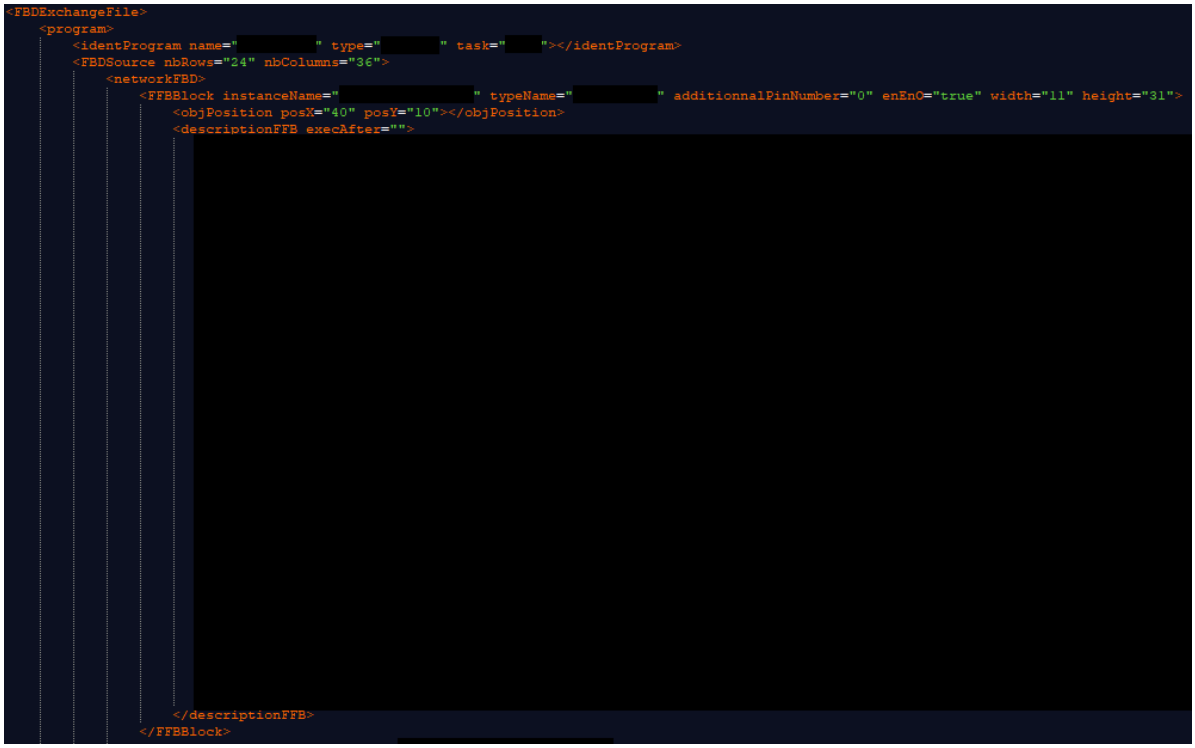
3.3 Järjestelmän rakenne

IO-lista-koodigeneraattori on osa suurempaa VEO Oy:n kehittämää Visual Studio projektihakemistoa, joka sisältää muita IO-listan käsittelyyn kehitettyjä työkaluja. Projektihakemistossa IO-lista-koodigeneraattorille on oma alikansio, jossa ohjelman tiedostot sijaitsevat. Ohjelma muodostuu neljästä toisiinsa kytköksissä olevasta pääkomponentista, jotka ovat: IOCodeGenerator.vb, Variable.vb, IOCodeGeneratorUI.xaml sekä IOCodeGeneratorUI.xaml.vb. Kansiorakenne kuvattuna alla (Kuvio 3).



Kuvio 3. Ohjelman kansiorakenne.

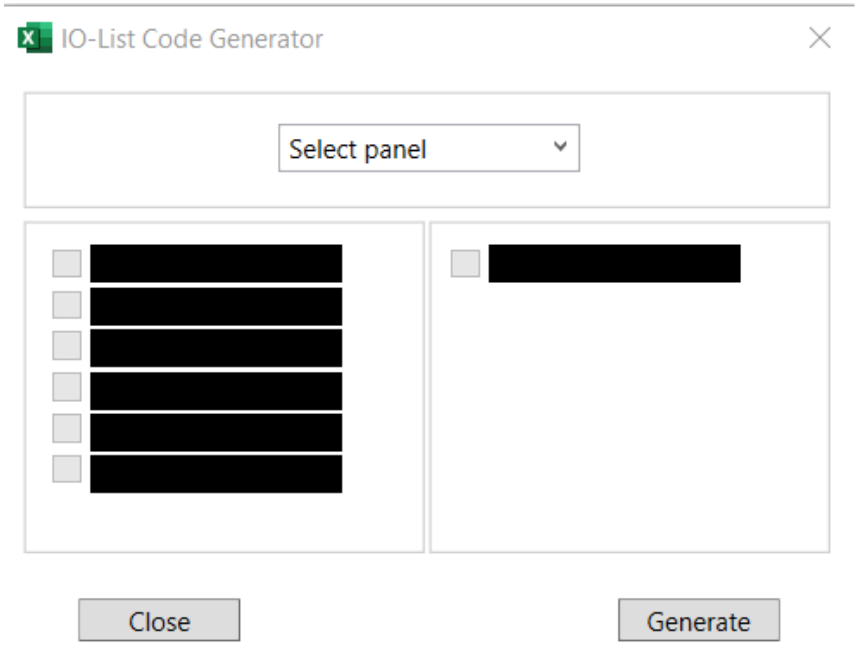
Ensimmäisenä mainittu luokka IOCodeGenerator.vb on ohjelman pääluokka, joka sisältää tarvittavat muuttujat sekä useita IO-listan sisällön käsittelyä ja tiedostoon kirjoitusta varten kehitettyjä funktio- sekä sub-aliohjelmia. Funktio- ja sub-aliohjelmat eroavat toisistaan siten, että funktiolla on palautusarvo, kun taas sub ei palauta mitään. IOCodeGenerator.vb luokassa luodaan paneelivalintojen perusteella tarvittava toimilohkokaavion, toimilohkoja sekä toimilohkojen kytkentöjä kuvaavan XML-tiedoston pohja. Toimilohkojen yksilökohtainen sisältö määräytyy IO-listan sisällön perusteella luotujen Variable-olioiden attribuuttien eli tietojen perusteella. Luokka Variable.vb sisältää tarvittavat muuttujat sekä aliohjelmat Variable-olion luomista varten. Valmis generoitu toimilohkokaavion kuvaava XML-tiedosto toimilohkoineen ja kytkentöineen havainnollistettu alla olevassa kuvassa (Kuvio 4). Generoitu XML-tiedosto tuotuna EcoStruxure Control Expert PLC-ohjelmointiohjelmaan ohjelman import-toiminnolla (Kuvio 5).



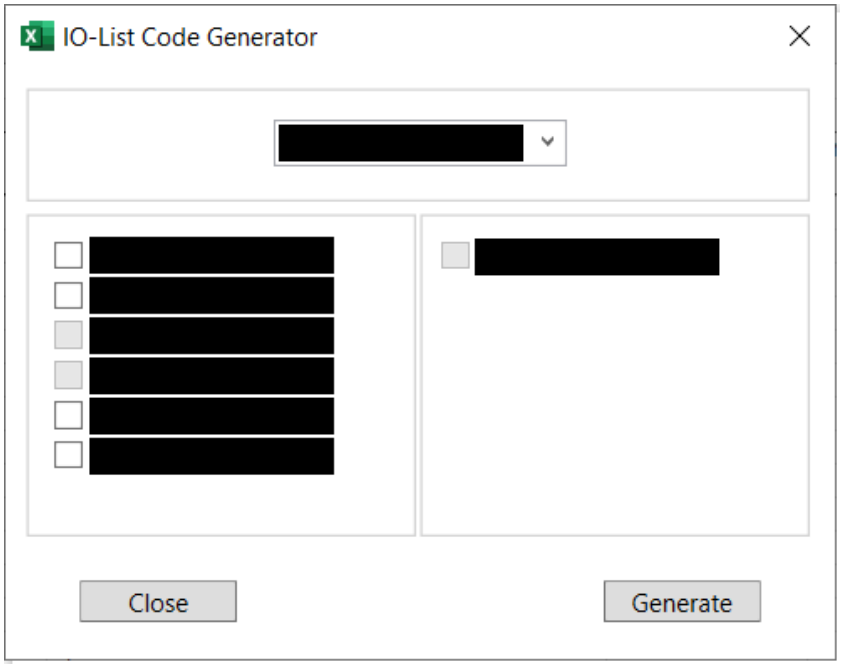
Kuvio 4. Esimerkkikuva ohjelmalla generoidun XML-tiedoston mahdollisesta sisällöstä (luottamukselliset tiedot piilotettu).

Kuvio 5. Generoitu toimilohkokaavion kuvaava XML-tiedosto tuotuna PLC-ohjelmointiohjelmaan (luottamukselliset tiedot piilotettu).

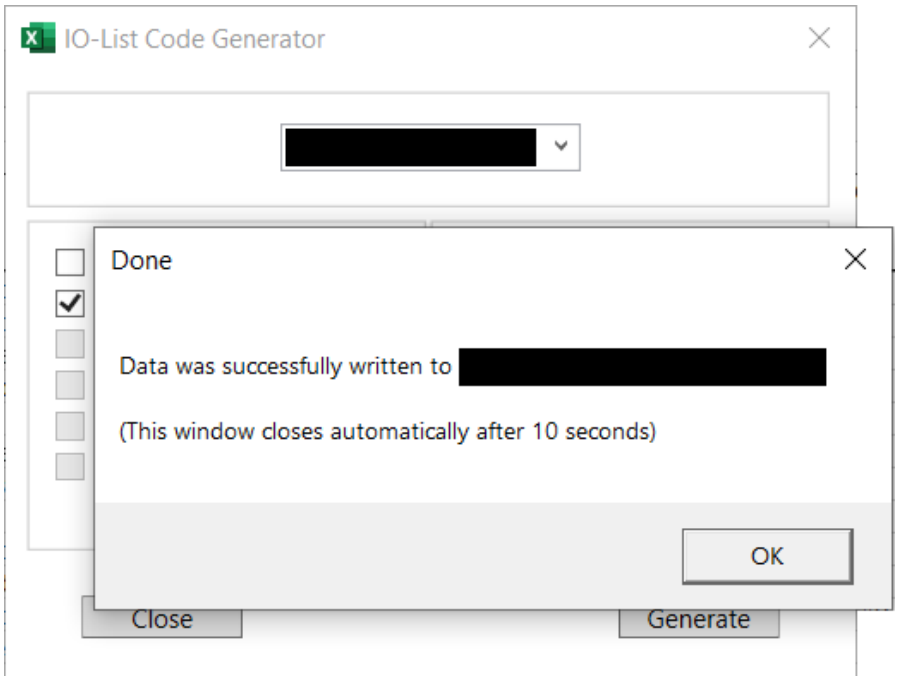
IOCodeGeneratorUI.xaml tiedostossa määritellään graafisen käyttöliittymän elementit sekä tapahtumat, kuten napin painalluksesta seuraava aliohjelman suorittaminen. IO-lista-koodigeneraattorin graafinen käyttöliittymä sisältää alasvetovalikon, josta voidaan valita haluttu paneeli, valintaruudut I/O-korteille sekä valintaruudun vaihtoehtoiselle kommunikaatioprotokollalle. Alkutilanteessa kaikkien valintaruutujen valitseminen on estetty ja vasta paneelin valitsemisen jälkeen sallitaan niiden valintojen tekeminen, mitä kullakin paneelilla on (Kuvio 6 ja Kuvio 7). Lisäksi käyttöliittymässä on nappi, josta sovelluksen voi sulkea sekä nappi, jonka painallus aloittaa XML-tiedoston generoinnin tehtyjen valintojen perusteella. Tiedoston generoinnin valmistuttua sovellus ilmoittaa prosessin onnistumisesta ponnahdusikkunassa (Kuvio 8). Luokka IOCodeGeneratorUI.xaml.vb sisältää tarvittavat muuttujat sekä aliohjelmat graafisen käyttöliittymän tapahtumien käsittelyyn.



Kuvio 6. Ohjelman alkutilanne ennen paneelin valitsemista (luottamukselliset tiedot piilotettu).



Kuvio 7. Paneeli valittu ja paneelilla olevien korttityyppien valinta sallittu (luottamukselliset tiedot piilotettu).



Kuvio 8. Tiedoston generoinnin onnistumisesta ilmoittava ponnahdusikkuna (luottamukselliset tiedot piilotettu).

4 TESTAUS JA KÄYTTÖÖNOTTO

4.1 Ohjelman testaus

Koska ohjelma toteutettiin siten, että sitä ohjelmoitiin aina yksi toiminnon toteuttava osa kerrallaan, päädyin testaamaan ohjelmaa aina uuden osan valmistuttua. Testaukset toteutin yksinkertaisesti kokeilemalla suorittaa ohjelmaa aina uusien lisäyksien jälkeen. Mikäli tiedostojen generointi onnistui halutulla tavalla, vertailin generoituja tiedostoja käytetyn IO-listan sisältöön varmistaakseni, että generoitujen tiedostojen sisältö vastasi IO-listan mukaisia tietoja. Mikäli testien tulokset olivat odotetun kaltaiset, siirryin seuraavaan vaiheeseen ja aloitin uuden osion ohjelmoinnin.

Varmistuakseni siitä, että ohjelman kehityksen ohella toteutettavan testauksen tuloksiin voisi luottaa, käytin uusia ominaisuuksia testatessa IO-listan eri versioita tiedostojen generoinnin pohjana. Tällä tavalla varmistin, että tekemäni ohjelma toimii tarkoituksenmukaisella tavalla eri tietoja sisältävien IO-listojen versioilla. Löysinkin tämän testaustavan avulla ohjelmasta joitakin ominaisuuksia, jotka toimivat yhdellä versiolla ja toisella taas ei. Huomattuani ohjelman puutteet pystyin korjaamaan löydetyt vikakohdat siten, että ohjelman ominaisuudet toimivat kaikissa IO-listan versioissa samankaltaisesti.

Ohjelman kehitysprosessin lähestyessä loppua, toiminnallisten ominaisuuksien valmistuttua lähetin kaikki ohjelmalla generoitavat tiedostot toimeksiantajayrityksen yhdyshenkilölle tarkasteltavaksi. Lisäksi loppuvaiheessa, kun toiminnalliset ominaisuudet sekä käyttöliittymän viimeistely oli valmistunut, toimeksiantajayrityksen projektin osalliset pääsivät itse testaamaan kehitettyä ohjelmaa käytännössä. Lähitulevaisuudessa ohjelmaa tullaan vielä toimeksiantajayrityksen puolesta testaamaan oikean projektin PLC-koodin generoinnissa.

4.2 Käyttöönotto

Ohjelman käyttöönottovaiheen ensimmäinen vaiheessa IO-lista-koodigeneraattori yhdistettiin projektiin, joka sisältää muut VEO Oy:n käyttämät IO-listan käsitte-

lyyn tarkoitetut työkalut. Yhdistämisen tarkoituksena on, että kaikki IO-listan käsittelytyökalut löytyvät sekä käynnistyvät samasta paikasta, joka tässä tapauksessa on Microsoft Exceliin asennettava Add-in -apuohjelma.

Asennettava Excel-apuohjelma luo uuden mukautetun välilehden Exceliin, josta työkaluja voidaan käyttää. Työkalujen yhdistämisprosessissa työssä kehitetyn ohjelman lähdekoodin sisältävät tiedostot kopioitiin muut työkalut sisältävään projektikansioon. Tiedostojen siirtämisen jälkeen IO-lista-koodigeneraattorille täytyi tehdä oma käynnistysnappi Excel-apuohjelman luomaan välilehteen. Koska IO-lista-koodigeneraattori yhdistettiin muiden työkalujen kanssa, se ei vaadi erillistä asennusta. Asennus tapahtuu yhteisen asennustiedoston avulla, joka asentaa kaikki IO-listan käsittelytyökalut yhteen kokoavan Excel-apuohjelman.

5 YHTEENVETO

Tämän opinnäytetyön tavoitteena oli kehittää VEO Oy:lle ohjelma, jonka avulla logiikkaohjelman tekoprosessia voidaan helpottaa ja nopeuttaa automatisoimalla tekoprosessi. Työn tuloksena syntyi ohjelma, joka täyttää kaikki ohjelman määrittelyssä sille asetetut vaatimukset ja toimii niiden mukaisesti. Työssä kehitetyllä ohjelmalla voidaan generoida raportin kirjoitushetkellä yhteensä 34 paneeli- ja korttivalintayhdistelmän mukaista käytettävän IO-listan sisällön perusteella määräytyvää yksilöllistä toimilohkokaaviota kuvaavaa XML-tiedostoa. Tositilanteessa kehitettyä ohjelmaa ei ole vielä käytetty, mutta lähitulevaisuudessa ohjelmaa tullaan käyttämään testiprojektissa logiikkaohjelman toimilohkojen generoinnissa.

Ohjelmalle asetetut vaatimukset täyttyivät työn aikana, mutta jatkokehitysmahdollisuuksia ohjelmasta löytyy kuitenkin edelleen. Ohjelmaa voisi laajentaa esimerkiksi kehittämällä siihen lisää generointimahdollisuuksia sekä yhdistämällä kaikki IO-listan käsittelytyökalut yhteisen käyttöliittymän alle. Lisäksi ohjelmaan voisi tarvittaessa lisätä generoinnin pohjana käytettävän IO-listan sekä generoinnin tuloksena syntyvän XML-tiedoston oikeellisuuden tarkastusmenetelmiä.

Tämän opinnäytetyön kehitysprosessi on ollut oppimisen kannalta erittäin hyödyllinen. Työn tekemisen tuloksena olen oppinut paljon uutta projektityöskentelystä sekä ohjelmistokehityksestä ja sen eri vaiheista. Työn aikana vastaan tulleista ongelmista selvittiin ja niiden ratkaisemiseksi täytyi keksiä sopivat keinot, mikä on kehittänyt ongelmanratkaisukykyjäni. Työn aihe oli mielenkiintoinen ja ajatus oikeasti tarpeellisen ohjelman kehittämisestä toimi hyvänä motivaattorina työtä tehdessä.

LÄHTEET

/1/ Holma, M. The first electrifying 25 years. Viitattu 10.2.2020. https://www.veo.fi/company/company-subpage/

/2/ Company. Viitattu 10.2.2020. https://www.veo.fi/company/

/3/ What is a PLC? Viitattu 13.2.2020. https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/what-plc/

/4/ What is VB.Net? Introduction, History, Features, Advantages, Disadvantages. Viitattu 31.5.2020. https://www.guru99.com/vb-net-introduction-features.html

/5/ Get started with WPF. Viitattu 31.5.2020. https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2019

/6/ What is WPF? Viitattu 31.5.2020. https://www.wpf-tutorial.com/about-wpf/what-is-wpf/